Appels système pour la réalisation des E/S

Selma - Département Informatique - IUT Fontainebleau/Sénart

Mécanisme des Entrées/Sorties.

Les octets de données d'un fichier que le processus veut lire (appel système *read)* sont copiés par le noyau depuis le *buffer cache* vers l'espace d'adressage du processus.

Les octets de données d'un fichier que le processus écrit (appel système *write*) sont copiés par le noyau de l'espace d'adressage du processus vers le *buffer cache*.

Le système d'exploitation (OS) transfert les blocs de données/méta-données entre les supports de stockage et la mémoire vive en fonction des demandes des processus.

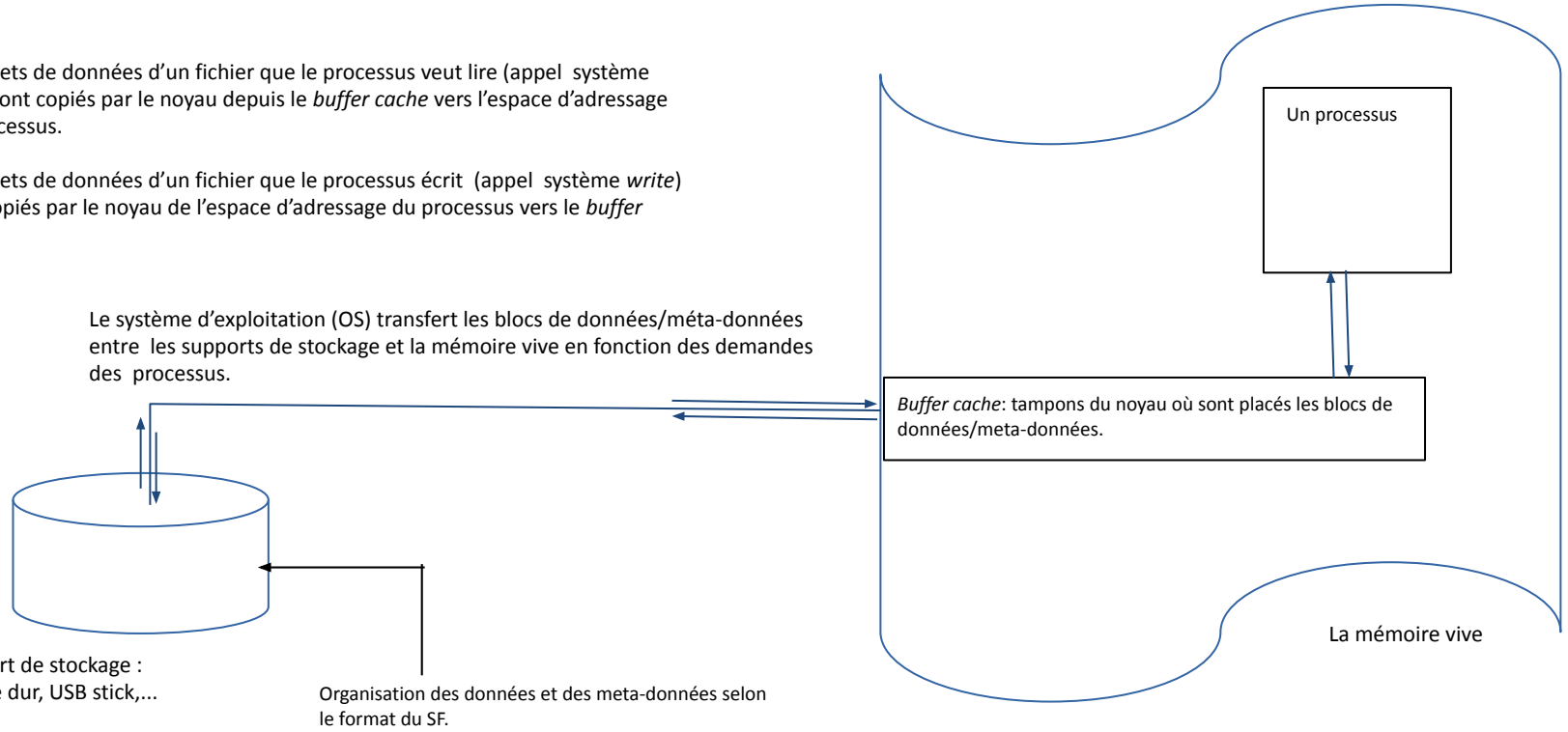


Table des fichiers ouverts du noyau et table des descripteurs de fichiers d'un processus.

Table des processus : une seule table. Elle est propriété du noyau.

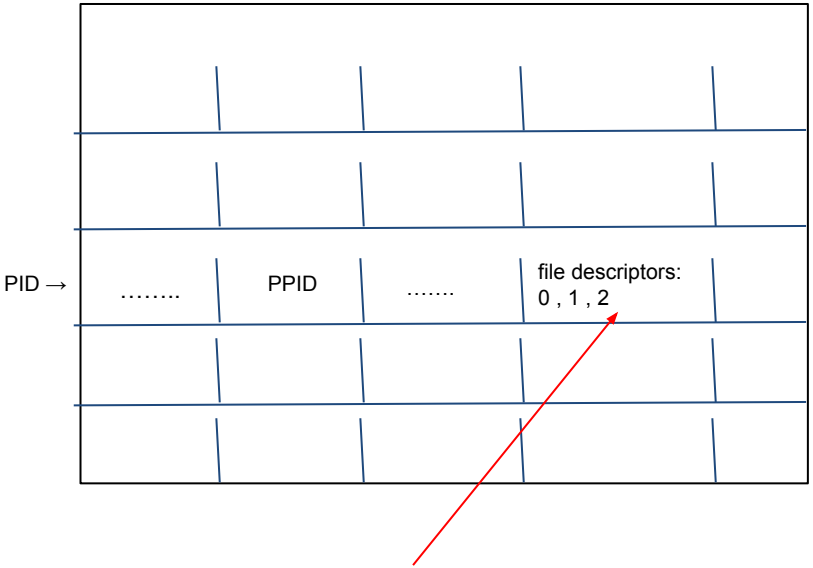
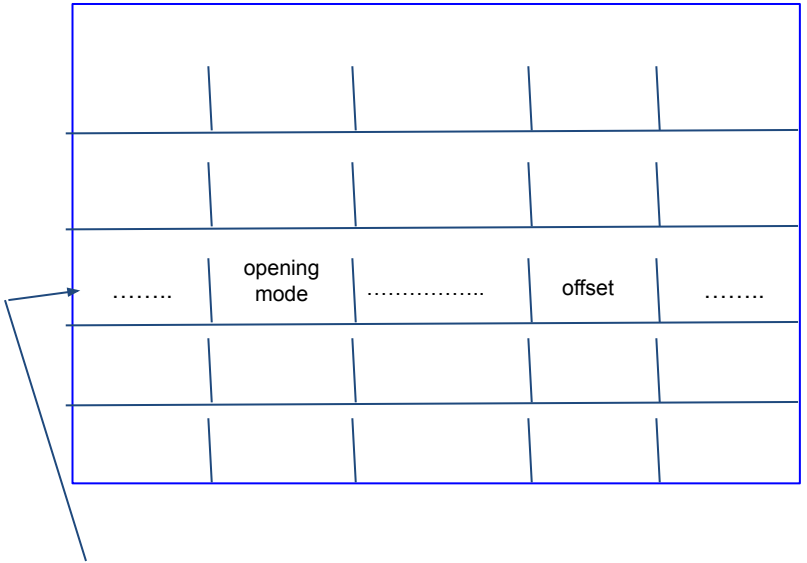


Table des descripteurs des fichiers ouverts par le processus. Les entrées 0,1,2 sont allouées automatiquement à la naissance du processus. Elles sont associées respectivement à l'entrée standard, et aux sorties standards résultats et erreurs du processus.

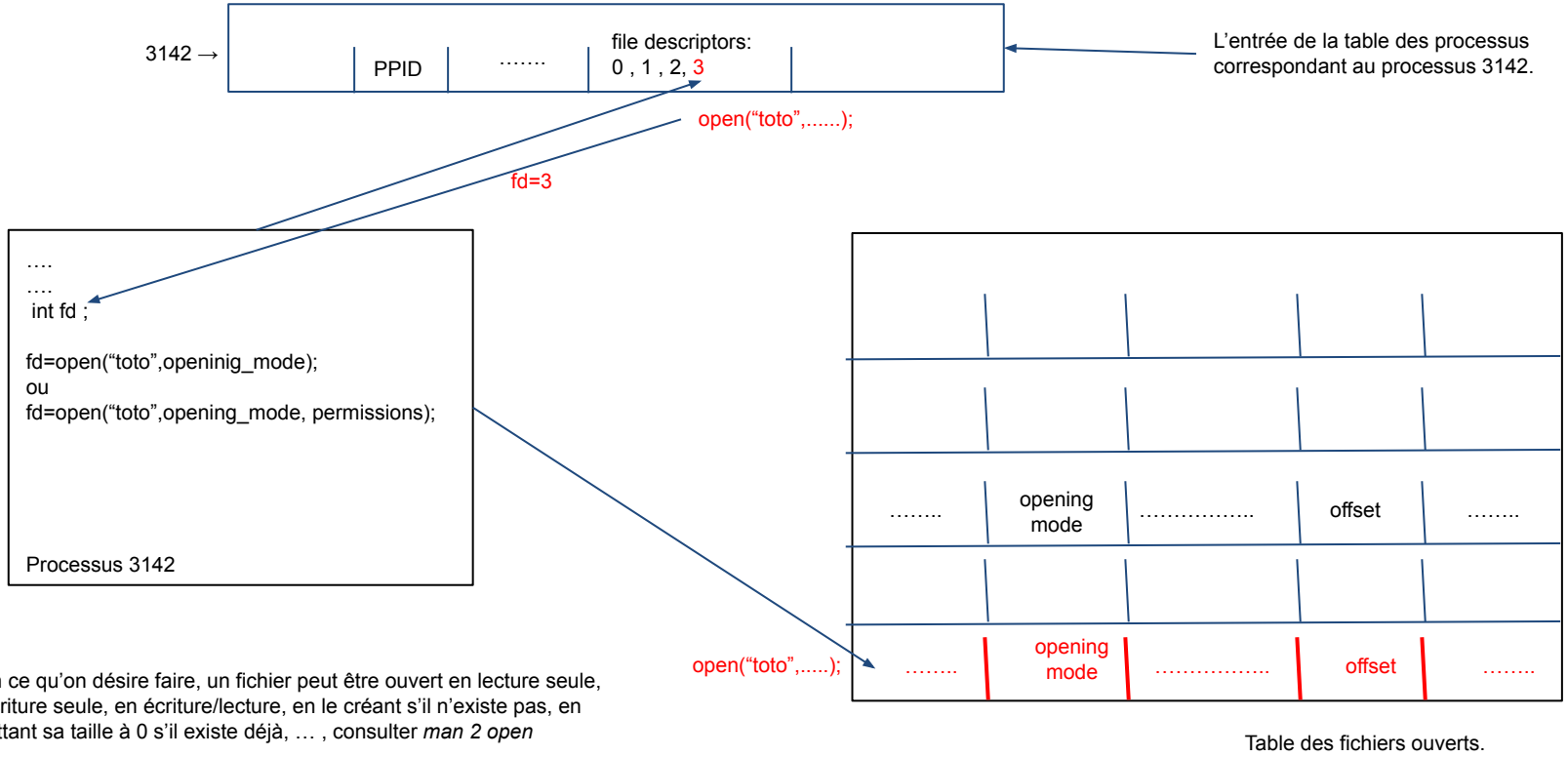
Table des fichiers ouverts : une seule table. Elle est propriété du noyau.



Chaque entrée de la table des fichiers ouverts enregistre les informations sur un fichier ouvert par un processus selon un mode d'ouverture. Un certain nombre d'informations sont enregistrées dans chacune de ces entrées comme le déplacement en octets (offset) dans le fichier.

Le noyau maintient le lien entre chaque entrée de la table des descripteurs de fichiers du processus et l'entrée correspondante dans la table des fichiers ouverts.

L'appel système open.



Selon ce qu'on désire faire, un fichier peut être ouvert en lecture seule, en écriture seule, en écriture/lecture, en le créant s'il n'existe pas, en remettant sa taille à 0 s'il existe déjà, … , consulter *man 2 open*

Table des fichiers ouverts.

L’appel système *read.*

$ man 2 read

```
READ(2)                          Linux Programmer's Manual                          READ(2)

NAME
       read - read from a file descriptor

SYNOPSIS
       #include <unistd.h>

       ssize_t read(int fd, void *buf, size_t count);

DESCRIPTION
       read()  attempts to read up to count bytes from file descriptor fd into the buffer start-
       ing at buf.
```

```
#define BFSZ 32

int fd; char buf[BFSZ];

fd=open(“pathname”,O_RDONLY); // Retourne -1 si echec.

n=read(fd,buf,BFSZ); // retourne -1 si échec. Sinon, retourne le nombre d’octets effectivement lus. Donc retourne 0 sur une
                                                                                                    // fin de fichier et pas -1.
```

Le processus demande d’obtenir dans son espace d’adressage une copie d’octets contenus dans le fichier “pathname”.

Le fichier est spécifié dans *read* par le descripteur de fichier préalablement retourné par le noyau à l’issue d’un appel système *open*. Le descripteur de fichier est spécifié dans le premier argument de *read.*

Les octets dont le noyau place une copie dans l’espace du processus sont consécutifs à partir du déplacement courant (offset) indiqué par le champ approprié dans l’entrée correspondante de la table des fichiers ouverts. Ici, on vient d’ouvrir le fichier. C’est donc à partir de l’offset 0.

Le nombre d’octets à copier dans l’espace du processus est spécifié par le processus dans le troisième argument de *read.*

Le processus doit indiquer au noyau une adresse dans son espace d’adressage où les octets doivent être placés. Cette adresse est spécifiée dans le deuxième argument de *read.*

A l’issue d’une opération *read r*éussie, l’offset est mis à jour par offset ← offset+n

L’appel système *write*.

C’est l’opération inverse de *read* : le processus veut que l’on écrive des octets de son espace d’adressage vers un fichier.

$ man 2 write

```
WRITE(2)                          Linux Programmer's Manual                          WRITE(2)

NAME
       write - write to a file descriptor

SYNOPSIS
       #include <unistd.h>

       ssize_t write(int fd, const void *buf, size_t count);

DESCRIPTION
       write()  writes  up to count bytes from the buffer starting at buf to the file referred to
       by the file descriptor fd.
```

```
int fd;

fd=open(“pathname”,O_WRONLY); // Retourne -1 si echec.
n=write(fd,”HELLO!\n”,7); // retourne -1 si echec. Sinon, retourne le nombre d’octets effectivement écrits.
```

Le processus demande que l’on écrive, dans le fichier “pathname”, les octets qui sont à l’adresse de son espace d’adressage indiquée dans le deuxième argument de *write*.

Le fichier est spécifié dans *write* par le descripteur de fichier préalablement retourné par le noyau à l’issue d’un appel système *open*. Le descripteur de fichier est spécifié dans le premier argument de *write*.

Les octets sont écrits dans le fichier au déplacement courant (offset) indiqué par le champ approprié dans l’entrée correspondante de la table des fichiers ouverts. Ici, on vient d’ouvrir le fichier. C’est donc à partir de l’offset 0.

Le nombre d’octets à écrire est spécifié par le processus dans le troisième argument de *write*.

A l’issue d’une opération *write* réussie, l’offset est mis à jour par offset ← offset+n

Se déplacer dans un fichier. L'appel système *lseek*.

Dans un fichier régulier, on peut aller à une position particulière en utilisant l'appel système *lseek*.

$ man 2 lseek

```
LSEEK(2)                    Linux Programmer's Manual                   LSEEK(2)

NAME
       lseek - reposition read/write file offset

SYNOPSIS
       #include <sys/types.h>
       #include <unistd.h>

       off_t lseek(int fd, off_t offset, int whence);

DESCRIPTION
       lseek()  repositions the file offset of the open file description asso-
       ciated with the file descriptor fd to the argument offset according  to
       the directive whence as follows:
```

Exemples. Rappel : les octets dans le fichier sont numérotés à partir de 0.

lseek(fd,32,SEEK_CUR); // L'offset dans le fichier correspondant au descripteur fd devient l'offset courant augmenté de 32.

lseek(fd,-64,SEEK_END); // L'offset dans le fichier correspondant au descripteur fd devient positionné sur le 64ème octet en partant de la fin du fichier.

lseek(fd,128,SEEK_SET); // L'offset dans le fichier correspondant au descripteur fd devient positionné sur l'octet numéro 128 en partant du début du fichier.

La valeur de retour est l'offset résultant mesuré en octets depuis le début du fichier, ou -1 en cas d'échec.